(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
31 May 2001 (31.05.2001)

PCT

(10) International Publication Number
WO 01/39012 A2

(51) International Patent Classification[7]: G06F 17/00

(21) International Application Number: PCT/US00/31951

(22) International Filing Date:
22 November 2000 (22.11.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/167,053 22 November 1999 (22.11.1999) US

(71) Applicant: AVENUE, A, INC. [US/US]; The Smith Tower, 9th floor, 506 Second Avenue, Seattle, WA 98104 (US).

(72) Inventor: SCHIPUNOV, Vladimir, Victorovich; 7557 20th Avenue NE, Seattle, WA 98115 (US).

(74) Agents: LAWRENZ, Steven, D. et al.; Perkins Coie LLP, P.O. Box 1247, Seattle, WA 98111-1247 (US).

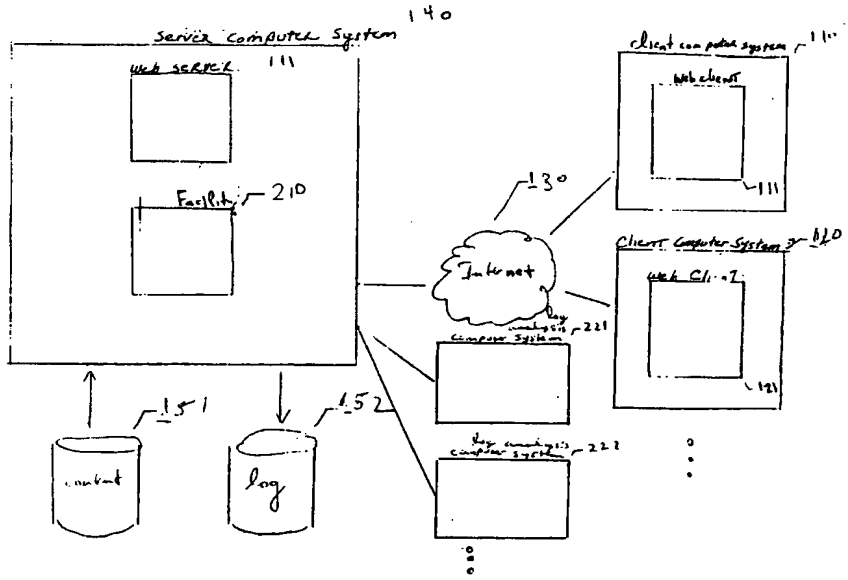
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: EFFICIENT WEB SERVER LOG PROCESSING



(57) Abstract: A facility for processing log entries is described. The facility receives a number of web serving requests. Each received request contains a unique identifier identifying the originator of the request. For each received request, the facility applies to the unique identifier contained by the request a mapping from unique identifiers to logs. By applying the mapping, the facility identifies a single log among a plurality of logs that is mapped from the contained unique identifier. The facility stores a log entry representing the web serving request in the identified log.

WO 01/39012 A2

EFFICIENT WEB SERVER LOG PROCESSING

TECHNICAL FIELD

The present invention is directed to data storage, retrieval, and processing techniques.

BACKGROUND

As computer use, and particularly the use of the World Wide Web, becomes more and more prevalent, the volume of traffic processed by web servers grows larger and larger.

Figure 1 is high-level block diagram the environment in which a web server operates. The diagram shows client computer systems, such as client computer systems 110 and 120. A user of client computer system 110 uses a web client 111, such as a web browser, to send an HTTP request via the Internet 130 to a server computer system 140. The HTTP request is directed to the server computer system by including a URL identifying the server computer system in the HTTP request. A web server program 141 executing in the server computer system processes the HTTP request by retrieving content referenced in the HTTP request from a content database 151 and returning it to the web client on the client computer system. Also in response to the HTTP request, the web server typically records information relating to the HTTP request in a web server log database 152.

By executing queries against the web server log database using a relational database engine, the contents of the log may be used to analyze the performance of the web server, or to discern the types of requests issued by client computer systems.

As the number of users using the World Wide Web grows, the rate at which a typical server computer system receives HTTP requests grows considerably. Accordingly, techniques for more efficiently storing, accessing, and analyzing web server log information would be of significant utility.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is high-level block diagram the environment in which a web server operates.

In some embodiments, the facility stores each record of the web server log in a condensed binary form to minimize the amount of space on the web server log occupied by each record. The facility preferably further compresses each record using a compression technique optimized for high-speed decompression.

By storing and analyzing log entries in this manner, the facility achieves new levels of efficiency in performing this logging and analysis function. Further, the facility is highly scalable, and provides powerful log analysis capabilities.

Figure 2 is a high-level block diagram that shows the environment in which the facility operates.

Figure 3 is a chart diagram showing the contents of a typical placement web server log record.

Figure 4 is a chart diagram showing the contents of a typical advertiser web server log record.

Figure 5 is a data structure diagram showing the human-readable contents of a group of typical element server log records.

Figure 6 is a data structure diagram showing the human-readable contacts of a group of typical advertiser server log records.

DETAILED DESCRIPTION

A software facility for efficiently storing and accessing web server logs ("the facility") is provided.

In certain embodiments, the facility distributes web server log entries across multiple log files to facilitate parallel reading and analysis of the web server log by a number of processors simultaneously. The facility achieves such distribution of web server log entries using a mapping that maps each new web server log entry into a single log file. In various embodiments, the mapping is based upon a unique identifier identifying a user that originated the web serving request to which the entry corresponds; the type of the web serving request to which the entry corresponds; and/or an effective time for the web serving request to which the entry corresponds. The facility also preferably stores web server log files without maintaining indices thereon, further reducing the processing requirements for storing the log. The facility preferably processes the logs using specialized analysis programs developed in an efficient general-purpose programming environment, rather than using a conventional relational database engine for such analysis. In some embodiments, these programs are written in C++, and call a standard template library.

In some embodiments, the facility stores each record of the web server log in a condensed binary form to minimize the amount of space on the web server log occupied by each record. The facility preferably further compresses each record using a compression technique optimized for high-speed decompression.

By storing and analyzing log entries in this manner, the facility achieves new levels of efficiency in performing this logging and analysis function. Further, the facility is highly scalable, and provides powerful log analysis capabilities.

Figure 2 is a high-level block diagram that shows the environment in which the facility operates. From this diagram, it can be appreciated that the facility 210 executes on the server computer system 140 within the environment described above in conjunction with Figure 1. The diagram also shows a number of log analysis computer systems, such as log analysis computer systems 221 and 222, that are connected to the server computer system 140, and that may be used by the facility to execute log analysis programs in parallel against different log partitions. Each log analysis computer system may have a single processor, or may have multiple processors.

While preferred embodiments are described in terms of the environment described above, those skilled in the art will appreciate that the facility may be implemented in a variety of other environments, including a single, monolithic computer system, as well as various other combinations of computer systems or similar devices.

To more fully illustrate its implementation and operation, the facility is described in conjunction with an example. The example relates to a specialized web server for managing and monitoring Internet advertising activity. In this specialized environment, the web server receives two types of HTTP requests: placement requests and advertiser requests. A placement request is sent when a user is visiting a web page that contains advertising, such as a banner advertising message. Retrieval of such a web page from its publisher by the user causes the client to issue a placement request to the web server to return the content of the advertising message. Placement requests are also received at the web server when the user clicks on the advertising message in order to browse to the web site of the advertiser. The web server receives an advertiser request when the user visits a particular page on the advertiser's web site. While the records stored in the web server log for placement and advertiser requests have several fields in common, they diverge in several respects.

Figure 3 is a chart diagram showing the contents of a typical placement web server log record. Each row of the chart corresponds to a different field defined for placement web server records. The Unix timestamp field in position one indicates the date and time that the placement request was received. In the example record, the Unix timestamp field, the Unix timestamp field contains the decimal value "921746335." This human-readable value is preferably stored in binary form as the hexadecimal value "36 F0 BB 9F."

The placement field in position two identifies the Internet publisher from which the placement request was received. In the example record, the human-readable placement label is "247_garden_030199sj_4." This human-readable value is preferably stored in binary form as the hexadecimal value "00 00 12 9C."

The advertising message ID field in position three indicates the particular advertising message to which the record relates. The example record has an advertising message ID of "27440," which is preferably stored in binary form as the hexadecimal value "00 00 6B 30."

The cookie field in the fourth position is an 8-byte value stored on the user's computer system to uniquely identify it to the web server. In the example record, the cookie is "918503847-16924218," preferably stored in binary form as hexadecimal value "36 BF 41 A7 01 02 3E 3A."

The IP address field in position five indicates the Internet protocol address at which the placement request originated. In the example record, the IP address field contains "130.166.82.106," which is preferably represented in binary form as the hexadecimal value "82 A6 52 6A." The hash value of user-agent field in position 6 contains a hashed indication of the user-agent, and is optional.

The flags field in position 7 contains flags identifying the specific nature of the advertise request, such as an "impression" flag. Placement log entries do not have an eighth field.

Figure 4 is a chart diagram showing the contents of a typical advertiser web server log record. The Unix timestamp field in position one indicates the date in time that the request was received. In the example record, the Unix timestamp field contains the value "942933252," which is preferably stored in binary form as the hexadecimal value "38 34 05 04."

The action tag field in position 2 indicates the nature of the action that the user took on the advertiser's web page. In the example record, the action tag has a value of "ubid_prd," which is preferably stored in binary form as the hexadecimal value "00 00 43 A1."

The third position contains a 4-byte field that is not used. The cookie field in the fourth position is an 8-byte value stored on the user's computer system to uniquely identify it to the web server. In the example record, the cookie contains the value "942120440-2699748," preferably stored in binary form as hexadecimal value "38 27 9D F8 00 29 31 E4."

The IP address field in position five indicates the Internet protocol address at which the placement request originated. In the example record, the IP address field contains the value "24.218.79.61," which is preferably represented in binary form as the hexadecimal value "18 DA 4F 3D."

The hash value of user-h in position 6 contains a hashed indication of the user-h and is optional.

The flags field in position 7 contains flags identifying the specific nature of the advertiser request, such as an "impression" flag. The flags field for the sample rows preferably includes both impression and action flags.

The extended data field in position 8 contains "extended data" further describing the user's action advertiser's web site. For example, the extended data field may contain coded indications of the amount of money that the user spent or the type of item that the user impressioned information about or purchased. The extended data is textual in nature, and preferably is

comprised of extended data pairs, separated by slashes that are of the form "[subfield name].[subfield data]." In the example, advertiser request, the extended data field contains the extended data pair "b.17," which indicates that subfield b has value 17.

Because extended data is of a textual nature, is of variable length, and is of potentially substantial length, when an advertiser request is received by the web server, the facility preferably stores two different log records to represent each advertiser request: a log record in binary form excluding the extended data field, and a log record in textual form including the extended data field. Retrieval operations where the contents of the extended data field are not of interest may be performed quickly against a log containing the binary version, while retrieval operation in which the contents of the extended data field are significant may be performed against the textual version that contains the extended data.

Figure 5 is a data structure diagram showing the human-readable contents of a group of typical element server log rows.

Figure 6 is a data structure diagram showing the human-readable contacts of a group of typical advertiser server log rows.

In order to compact the contents of each web server row, the facility preferably compresses each row with a compression algorithm optimized for fast decompression. In a preferred embodiment, the facility utilizes a variant of LEMPEL-ZIV-OBERHUMER ("LZO") compression algorithm described at:

HTTP://wildshu.idv.uni-linz.ac.at/mfx/lzodoc.html

In particular, the facility preferably uses the LZO1X-1 variant of the LZO compression algorithm.

Some embodiments of the facility partition the log into a number of different log files, also called log partitions. The log files are partitioned

based upon one or more of the following bases: a time period in which the effective time of each entry in the log file is contained; one or more log entry types that the log file contains; and the range of unique identifiers contained by entries in the log. The time period basis for partitioning the logs results in different sets of logs for different periods of time, such as each hour, each day, or each month. The entry type basis for partitioning the log produces different sets of log files for different entry types. This permits many log files to utilize fixed-length entries of the minimum necessary size, and minimizes the number of logs that must be read when analyzing entries of fewer than all of the types. The unique identifier basis for partitioning the log minimizes the number of log files that must be read in order to analyze the behavior of a single user.

Partitioning the log into separate files in this manner facilitates parallel processing of the log. For example, in performing an analysis of a portion of the log corresponding to 50 log files, the facility may distribute the processing of each of the 50 log files to a different processor, or to a different computer system.

Rather than analyzing the contents of the log and log files conventionally by submitting queries to a relational database engine, the facility preferably uses programs constructed in a general-purpose programming environment to directly read and analyze the log as partitioned into log files. In particular, embodiments of the facility utilize the Standard Template Library (STL) now provided as part of the C++ programming language to provide this functionality. The STL, which is well known to those skilled in the art, is described succinctly in the document "An Overview of The Standard Template Library" by G. Bowden Wise, currently available at http://www.cs.rpi.edu/~wiseb/xrds/ovp2-3b.html, and is more formally defined in chapters 23-25 of International Standard ISO/IEC 14882 for "Programming Languages – C++", adopted by the American National Standards Institute on July 27, 1998 and adopted by the International Standardization Organization on September 1, 1998, currently available at http://webstore.ansi.org/.

The STL provides an orthogonal component structure, in which data is collected in containers, accessed there by iterators, and thereby provided to algorithms for processing. In order to perform analysis on a log file, the facility first reads the log file into a container such as a vector so that each entry in the log file occupies an item in the vector. The facility sorts the vector first by unique identifier, then by effective time. The facility then uses an iterator to iterate through the sorted vector in order. Because the vector is sorted first by unique identifier, all of the entries relating to a particular unique identifier occur contiguously in the sorted vector. Because the vector is sorted second by effective time, these contiguous entries relating to a single unique identifier occur in the order of their effective times. Thus, an algorithm receiving data from an iterator iterating through the sorted vector receives log entries for each unique identifier in turn. For a particular unique identifier, the entries are received in the order of their effective times. The algorithm utilized by the facility performs the desired analysis, such as analyzing a correspondence between a web serving request for placing a product in a shopping cart, and a later web serving request for authorizing payment for the product. Those skilled in the art will appreciate that algorithms performing virtually any type of analysis may be incorporated in the facility.

It will by understood by those skilled in the art that the above-described facility could be adapted or extended in various ways. For example, the facility may log and analyze transactions of various types other than web serving transactions. Also, log contents may be analyzed using tools other than C++ programs calling the standard template library. While the foregoing description makes reference to preferred embodiments, the scope of the invention is defined solely by the claims that follow and the elements recited therein.

CLAIMS

1. A method in a computing system for processing log entries, comprising:

receiving a plurality of web serving requests, each web serving request containing a unique identifier identifying an originator of the request;

for each received web serving request:

applying to the unique identifier contained by the request a mapping from unique identifiers to logs to identify a single log among a plurality of logs that is mapped to from the contained unique identifier; and

storing a log entry representing the web serving request in the identified log.

2. The method of claim 1 wherein each unique identifier received in a web serving request is a cookie retained by the originator of the web serving request.

3. The method of claim 1 wherein the mapping maps each of a plurality of disjoint ranges of identifiers to a different log.

4. The method of claim 1 wherein no index updates are performed to reflect the storage of log entries.

5. The method of claim 1 wherein each web serving request is of one of a number of types, and wherein the applied mapping is from unique identifier and request type to logs.

6. The method of claim 1 wherein each web serving request has an effective time, and wherein the applied mapping is from unique identifier and effective time to logs.

7. The method of claim 1, further comprising analyzing the contents of two or more of the plurality of logs without utilizing a relational database management system.

8. The method of claim 1, further comprising analyzing the contents of two or more of the plurality of logs under the control of a program constructed using a general-purpose programming technique.

9. The method of claim 1, further comprising analyzing the contents of two or more of the plurality of logs under the control of a C++ program calling a standard template library.

10. The method of claim 1, further comprising analyzing the contents of two or more of the plurality of logs each in a different processor.

11. The method of claim 1, further comprising:

receiving a request to process a subset of the plurality of logs;

in response to receiving the request to process, for each of the subset of the plurality of logs:

producing a version of the log that is sorted by unique identifier;

traversing the sorted version of the log, analyzing in turn sets of entries where each set of entries represents web serving requests originated by a different originator.

12. The method of claim 11 wherein each web serving request has an effective time contained in the log entry representing the web serving request, and wherein the produced version of the log is sorted first by unique identifier, then by effective time, and wherein the entries representing web serving requests originated by each originator are encountered in the traversal in the order that they occurred.

13. The claim 1 wherein each stored log entry contains the unique identifier read for the web serving request represented by the stored log entry, further comprising:

receiving a request to process a subset of the plurality of logs;

in response to receiving the request to process, for each of the subset of the plurality of logs:

reading the log entries stored in the log into a vector;

sorting the vector based upon the unique identifier of each entry; and

traversing the sorted vector, analyzing entries representing web serving requests originated by each originator in turn.

14. The method of claim 11 wherein a sorted version of the log is produced by sorting the log.

15. The method of claim 11 wherein a sorted version of the log is produced by extracting from the log the entries contained by the log, then sorting the extracted entries.

16. A computer-readable medium whose contents cause a computing system to processing web serving requests by:

receiving a plurality of web serving requests, each web serving request containing a unique identifier identifying an originator of the request;

for each received web serving request:

applying to the unique identifier contained by the request a mapping from unique identifiers to logs to identify a single log among a plurality of logs that is mapped to from the contained unique identifier; and

saving a log entry representing the web serving request in the identified log.

17. A computing system for processing log entries, comprising:

a web server that receives a plurality of web serving requests, each web serving request containing a unique identifier identifying an originator of the request;

a logging subsystem that, for each received web serving request:

applies to the unique identifier contained by the request a mapping from unique identifiers to logs to identify a single log among a plurality of logs that is mapped to from the contained unique identifier; and

records a log entry representing the web serving request in the identified log.

18. The computing system of claim 17, further comprising:

an analysis request receiver for receiving request to perform log analysis, each analysis request implicating one or more of the plurality of logs; and

for each log implicated by a received analysis request, a separate processor for performing on the log analysis specified by the received analysis request.

19. One or more computer memories that collectively contain a transaction logging data structure representing a plurality of transactions, each transaction having an originator among a domain of originators and a time among a domain of times, the data structure comprising a plurality of differentiated stores, each store containing transaction records representing transactions whose originators are within a range of originators associated with the store and whose times are within a range of times associated with the store,
such that all of the transaction records for representing transactions originated by each originator are contained by a minority of the stores.

20. The computer memories of claim 19 wherein each transaction is of one of a plurality of types, and wherein the transaction records contained by each store represent transactions each of one of a proper subset of the plurality of types that is associated with the store.

21. The computer memories of claim 19 wherein each transaction record contained by a store represents a web serving transaction.

22. A method in a computer system for managing web server logs, comprising:
for each of a plurality of web hits, generating a new binary log entry containing information about the web hit; and
for each new log entry,
selecting one of a plurality of logs to which to add the new log entry; and

adding the new log entry to the selected log.

23. The method of claim 22 wherein the log entries are ordered chronologically.

24. The method of claim 22 wherein the log files are not the subject of an index.

25. The method of claim 22, further comprising, before adding the new log entry to the selected log, compressing the new log entry.

26. The method of claim 22 wherein a LEMPEL-ZIV-OBERHUMER compression technique is utilized.

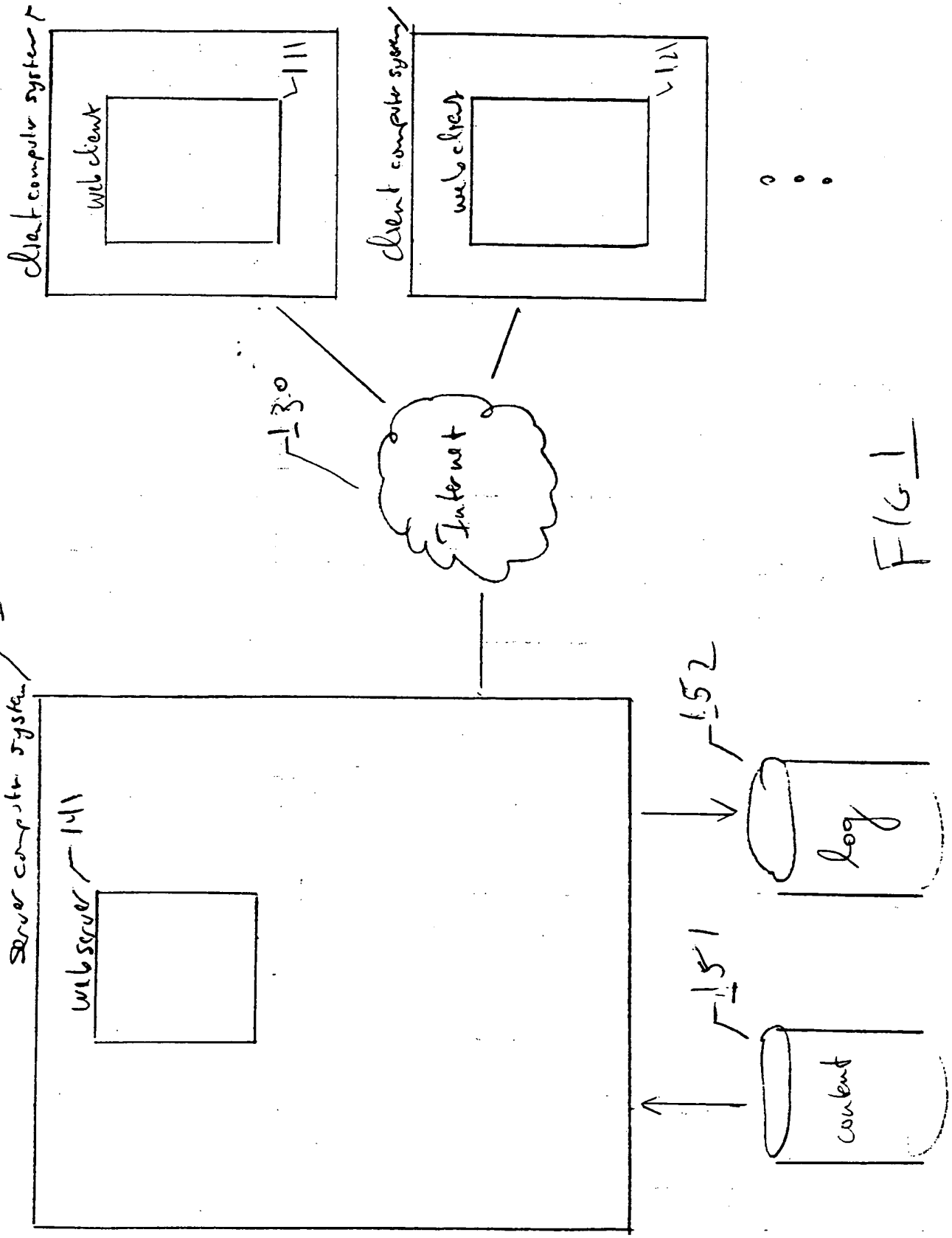


FIG 1

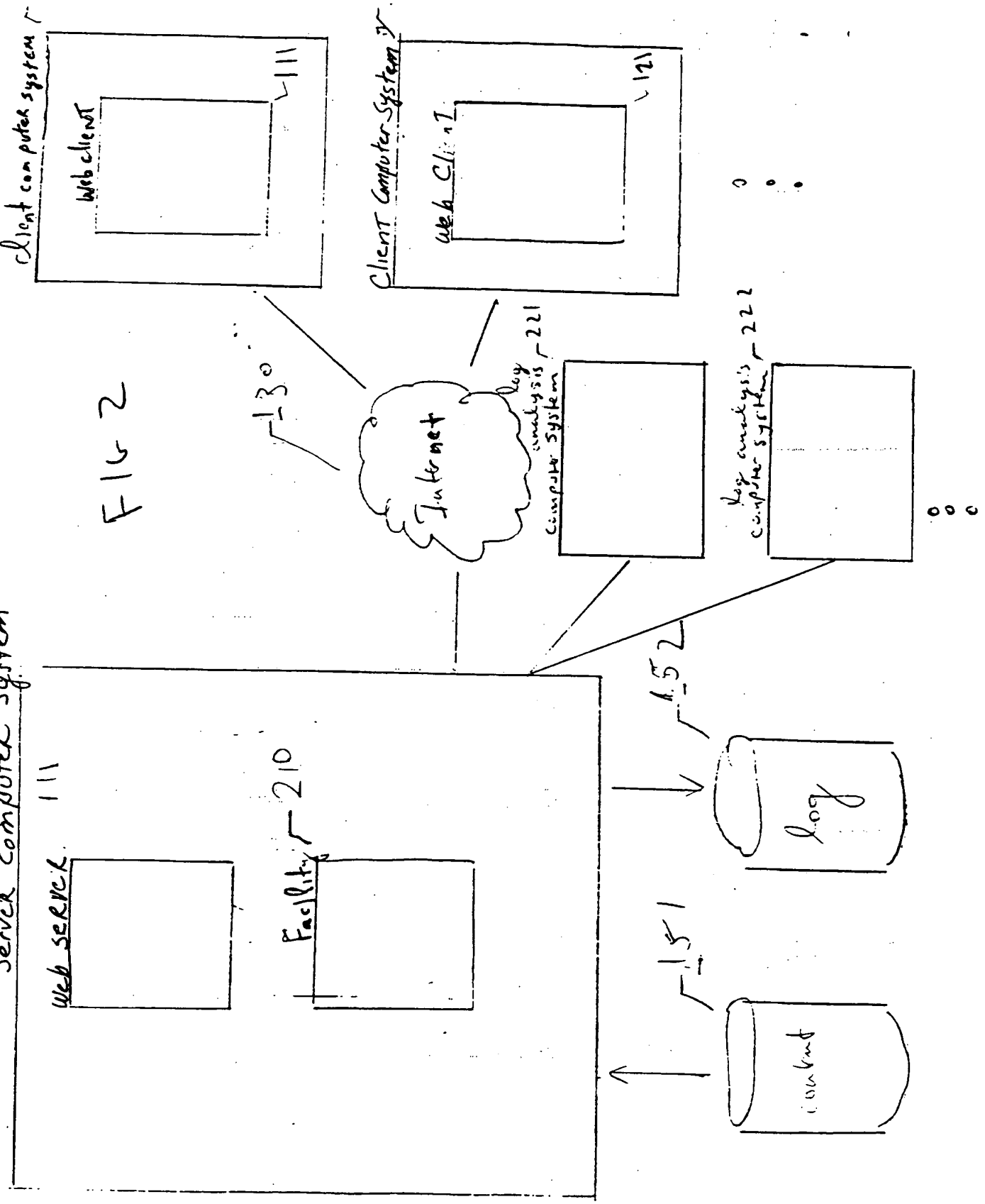


FIG 2

TRK log entry

Position	field	binary field size	sample content (human readable)	sample content (binary)
1	UNIX timestamp	4 bytes	921746335	36 F0 BB 9F
2	Placement	Integer ID, 4 bytes	247_garden_030199sj_4	00 00 12 9C
3	Advertisement ID	Integer ID, 4 bytes	27440	00 00 6B 30
4	Cookie	8 bytes (64 bits) unique ID	918503847-16924218	36 BF 41 A7 01 02 3E 3A
5	IP address	4 bytes	130.166.82.106	82 A6 52 6A
6	Hash value of User-Agent (optional)	4 bytes	0	00 00 00 00
7	Flags	12 bytes	-impression	00 00 00 00 00 00 00 00 00 00 00 00
8	Not used	N/A	0	N/A

FIG 3

ACT log entry, w/ extended data

Position	field	binary field size	sample content (human readable)	sample content (binary)
1	UNIX timestamp	4 bytes	942933252	38 34 05 04
2	Action tag	Integer ID, 4 bytes	ubid_prd	00 00 43 A1
3	Not used	4 bytes	0	00 00 00 00
4	Cookie	8 bytes (64 bits) unique ID	942120440-2699748	38 27 9D F8 00 29 31 E4
5	IP address	4 bytes	24.218.79.61	18 DA 4F 3D
6	Hash value of User-Agent (optional)	4 bytes	0	00 00 00 00
7	Flags	12 bytes	-action-ubid_prd-- impression-	00 00 00 00 00 00 00 00 00 00 00 00
8	Extended data	N/A	b.17	N/A

FIG 4

921746335	247_garden_030199sj_4	27440	918503847-16924218	130.166.82.106	0	impression
921746335	unique_garden_031099sj_1	27434	921746335-16783482	200.239.123.160	3664427135	impression 00-
921746335	gotocom	28318	921263039-16846378	208.162.6.218	0	impression
921746335	247_fogdog_022699eb_1	27624	921743698-16808999	24.4.91.180	0	impression
921746335	cybereps_proflowers_030499sj_1	27585	920630454-16843795	203.24.115.146	0	impression
921746335	cyberreps_bestprice_030499eb_1	27611	905802927-16805046	193.232.88.16	0	impression
921746335	cybereps_proflowers_030499sj_1	27585	921746335-16853646	152.163.232.25	1584920321	impression 00-
921746335	excite_uproar_012999sj_2	26696	921746335-16916158	194.152.172.3	3979746404	impression 00-

FIG 5

942933252	ubid_prd	0	942120440-2699748	24.218.79.61	0	-action ubid_prd--impression-	b.17
942933252	ubid_prd	0	942680525-1119807	205.217.107.121	0	-action ubid_prd--impression	b.9
942933252	ubid_bdcn	0	913061175-287141	207.88.159.68	0	-action ubid_bdcn--impression-	a.17055191/b./c.303832
942933254	ubid_prd	0	942932994-215798	63.14.35.146	0	-action ubid_prd--impression-	b.41
942933254	ubid_prd	0	911484025-147159	205.142.0.72	0	-action ubid_prd--impression-	b.2

FIG 6